Implicit sampling for path integral control, Monte Carlo localization and SLAM

Matthias Morzfeld
Department of Mathematics
University of California, Berkeley,
and
Lawrence Berkeley National Laboratory

Abstract

Implicit sampling is a recently-developed variationally-enhanced sampling method, that guides its samples to regions of high probability, so that each sample carries information. Implicit sampling may thus improve the performance of algorithms that rely on Monte Carlo methods. Here the applicability and usefulness of implicit sampling for improving the performance of Monte Carlo methods in estimation and control is explored, and implicit sampling based algorithms for stochastic optimal control, stochastic localization, and simultaneous localization and mapping (SLAM) are presented. The algorithms are tested in numerical experiments where it is found that fewer samples are required if implicit sampling is used, and that the overall runtimes of the algorithms are reduced.

1 Introduction

Many problems in physics and engineering require that one produces samples of a random variable with a given probability density function (pdf) p [5, 10]. If p is difficult to sample, one can sample an easier-to-sample pdf p_0 (the importance function), and attach to each sample $X^j \sim p_0$, $j = 1, \ldots, M$, the weight $W^j \propto p(X^j)/p_0(X^j)$ (capital letters denote realizations of random variables, and these realizations are indexed in superscript; subscript indices label discrete time). The weighted samples X^j form an empirical estimate of the pdf p and, under mild assumptions, the empirical estimate converges weakly to p, i.e., expected values of a function h with respect to p can be approximated by weighted averages of the function values $h(X^j)$. The difficulty is to find a "good" importance function p_0: if p_0 is not a good approximation of p, for example, if p_0 is not large when p is large, then many of the samples one proposes using p_0 may have a small probability with respect to p and, therefore, carry little information, and make sampling inefficient.

Implicit sampling is a general method for constructing useful importance functions [6, 7]. The basic idea is to first locate the region of high probability with respect to p via numerical optimization, and then to map a reference variable to this region; this mapping is done by solving algebraic equations. While the cost of generating a sample with implicit sampling is larger than in many other Monte Carlo (MC) schemes, implicit sampling can be efficient because the samples carry more information so that fewer samples are required. In [2, 29, 30]

this trade-off was examined in the context of geophysics, where it was found that implicit sampling indeed can be efficient, in particular when the dimension of the problem is large.

Here implicit sampling is applied to estimation and control in robotics, and new algorithms for stochastic optimal control, Monte Carlo localization, and simultaneous localization and mapping (SLAM) are developed. Implementations and efficiencies of these algorithms are illustrated and explored with examples. In particular, it is investigated if implicit sampling, which requires fewer samples that are however more informative and more expensive, can be efficient compared to other sampling schemes that may require more samples, each of which is less informative, however cheaper to generate.

The optimal control of a stochastic control problem can be found by solving a stochastic Hamilton-Jacobi-Bellman (HJB) equation [37]. The dimension of the domain of this partial differential equation (PDE) equals the dimension of the state space of the control problem. Classical PDE solvers require a grid on the domain and, therefore, are impractical for control problems of moderate or large dimension. For a class of stochastic optimal control problems, one can use MC solvers instead of grid based PDE techniques and, since the MC approach avoids grids, it is in principle feasible to solve larger dimensional control problems within this class [21–23, 38]. However, the sampling scheme must be chosen carefully or else MC based PDE solving will also fail (in the sense that unfeasibly many samples are required). It is shown in section 3 how to apply implicit sampling to obtain a practical algorithm for stochastic optimal control that avoids many of the pitfalls one faces in MC based PDE solving. The method and algorithm are illustrated with the double slit problem (see section 3.1 and [21]), which is a simple but vivid example of how things can go wrong, and how they can be fixed with implicit sampling. Using an inverse dynamics controller for a two-degrees-of-freedom robotic arm as an example, it is also shown how to obtain a semi-analytic solution for a linear Gaussian problem via implicit sampling. Finally, it is indicated how implicit sampling and stochastic optimal control can help with being trapped in local minima of non-convex optimization problems. An extension of the method presented here is also discussed in the conference paper [46].

In robotics, one often updates the predictions of a dynamic model of an autonomous robot with the output of the robot's sensors (e.g. radar or lidar scans). This problem is often called "localization", and can be formulated as a sampling problem. Localization algorithms that rely on MC sampling for the computations are called Monte Carlo localization (MCL) [8]. One can also learn the geometry and configuration of the map while simultaneously localizing the robot in it, which leads to the problem of "simultaneous localization and mapping". Efficient numerical solutions of the MCL and SLAM problems are a fundamental requirement for autonomous robots [26, 40]. The various MCL and SLAM algorithms differ in their importance function p_0, and some algorithms have been shown to be inefficient due to a poor choice of p_0 [26]. Here it is shown how to use implicit sampling to generate an efficient importance function for MCL and SLAM. The implicit sampling based MCL and SLAM algorithms are convergent, i.e. as the number of samples goes to infinity one obtains an empirical estimate of the true posterior even if the underlying dynamics are nonlinear (whereas many other SLAM algorithms require linearity for convergence [26]). The memory

requirements of the implicit sampling based SLAM algorithm scale linearly with the number of features in the map. The efficiencies of the new MCL and SLAM algorithms are compared to the efficiencies of competing MCL and SLAM algorithms in numerical experiments with the data set [32].

2 Review of implicit sampling

The efficiency of MC sampling depends on how well the importance function p_0 approximates the target pdf p. In particular, p_0 must be large where p is large, or the samples one produces using p_0 are unlikely with respect to p. Implicit sampling is a general MC sampling technique that constructs an importance function that is large where p is large by mapping a reference variable to the region where p is large [2, 6, 7, 29, 30]. Here implicit sampling is described in general terms; below implementations of implicit sampling are described in the context of applications.

The region where p is large is the region where $-\log(p)$ is small (the logarithm is used here because in applications p often involves exponential functions). The region where p is large can thus be located via minimization of the function

$$F(x) = -\log\left(p\left(x\right)\right).$$

If F is convex, the minimizer $\mu = \text{argmin } F$ (i.e. the mode of p) is the most likely value, and high-probability samples are in its neighborhood. One can obtain samples in this region by solving the stochastic algebraic equation

$$F(x) - \phi = \frac{1}{2}\xi^T\xi, \tag{1}$$

where $\phi = \min F$, $\xi \sim \mathcal{N}(0, I_m)$ is an m-dimensional Gaussian reference variable and where T denotes a transpose and I_m is the identity matrix of order m; here, and for the remainder of this paper, $\mathcal{N}(a, B)$ is shorthand notation for a Gaussian pdf with mean a and covariance matrix B. Note that the right-hand-side of (1) is likely to be small because ξ is close to the origin, which implies that the left-hand-side is small, which in turn means that the solution of (1), i.e. the sample, is close to the mode and, thus, in the high-probability region.

The weights of the samples are proportional to the absolute value of the Jacobian determinant of the map that connects the sample X^j to the reference variable ξ:

$$W(X^j) \propto \left|\det\left(\frac{\partial x}{\partial \xi}(X^j)\right)\right|.$$

In practice, the weights are usually normalized so that their sum is one. Various ways of constructing a map from ξ to x have been reported in the literature [6, 30] two of which are summarized below. In summary, implicit sampling requires (i) minimizing $F = -\log p(x)$; and (ii) solving equation (1). This two-step approach makes efficient use of the available computational resources: the minimization identifies the high probability region and the

samples are focused to lie in this region, so that (almost) all samples carry information. This is in contrast to other sampling schemes where an importance function is chosen ad-hoc, which often means that many of the samples carry little or no information; the computations used for generating uninformative samples are wasted.

Finally, the assumption that F is convex can be relaxed. Implicit sampling can be used without modification if F is merely U-shaped, i.e. the target pdf, p, is unimodal. Multimodal target pdfs can be sampled e.g. by using mixture models, for which one approximates each mode using the above recipe (see [6, 30, 45] for more detail).

2.1 Generating samples with random maps

To generate samples, one solves (1) with a one-to-one and onto mapping. There are many choices for such a mapping, and one is to solve (1) in a random direction, i.e. one puts

$$X^j = \mu + \lambda^j L^{-T} \eta^j, \tag{2}$$

where L is a Cholesky factor of the Hessian of F at the minimum, $\eta^j = \xi^j / \sqrt{(\xi^j)^T \xi^j}$ is a vector which is uniformly distributed on the m-dimensional hypersphere, and where λ^j is a scalar. One then computes λ^j by substituting (2) into (1). This approach, called "random map", requires solving a scalar equation to generate a sample [30]. Moreover, the Jacobian of this map is easy to evaluate with the formula (see [30] for a derivation)

$$\left| \det \left(\frac{\partial x}{\partial \xi} \right) \right| = \frac{(\xi^T \xi)^{(1-m)/2}}{\det(L)} \left| \frac{\lambda(\xi)^{m-1}}{2 \nabla F \; L^{-T} \eta} \right|, \tag{3}$$

where ∇F is the gradient of F with respect to x (an m-dimensional row vector).

In numerical implementations of this method, calculating ∇F may require repeated evaluations of F, e.g. if the gradient is approximated via finite differences. This can be avoided by noting that

$$\frac{d\lambda}{d\rho} = \frac{1}{2 \nabla F \; L^{-T} \eta},$$

where $\rho = \xi^T \xi$, so that the Jacobian becomes

$$\left| \det \left(\frac{\partial x}{\partial \xi} \right) \right| = \frac{\rho^{(1-m)/2}}{\det(L)} \left| \lambda(\xi)^{m-1} \frac{d\lambda}{d\rho} \right|. \tag{4}$$

The scalar derivative $d\lambda/d\rho$ can be evaluated using finite differences with a few evaluations of F (the precise number of function evaluations depends on the finite difference scheme one chooses).

2.2 Generating samples with approximate quadratic equations

Another path to solving (1) is to replace it with an approximate quadratic equation

$$\hat{F}(x) - \phi = \frac{1}{2} \xi^T \xi, \tag{5}$$

where

$$\hat{F}(x) = \phi + \frac{1}{2}(x-\mu)^T H (x-\mu),$$

is the Taylor expansion of order two of F about its minimizer μ, and where H is the Hessian of F at the minimum. A solution of (5) is

$$X^j = \mu + L^{-T}\xi_j,$$

where L is the Cholesky factor of $H = LL^T$. The weights are

$$W^j \propto \exp\big(-\phi + \hat{F}(X^j) - F(X^j)\big), \tag{6}$$

and account for the error one makes by solving (5) instead of (1) (see [6] for more detail).

3 Application to path integral control

The finite horizon stochastic optimal control problem considered here is as follows: find a control u (a p dimensional vector function of the state x) that minimizes the cost function

$$C(x_0, t_0, u) = E\left[\Phi(x(t_f)) + \int_{t_0}^{t_f} u(x(\tau),\tau)^T R u(x(\tau),\tau) + V(x(\tau),\tau) d\tau\right],$$

where x is an m-dimensional vector (the state), $t_0 \le t \le t_f$ is time, t_f is the final time (or the horizon), Φ is a scalar function that describes the "final cost", R is a $p \times p$ symmetric positive definite matrix that specifies the control cost, and V is a scalar function which describes the "running cost" (which is also called "potential"); the expectation is taken over trajectories of the stochastic differential equation (SDE)

$$dx = f(x,t)\,dt + G\,(u\,dt + Q\,dW), \tag{7}$$

starting at $x(t_0) = x_0$, where f is a smooth m-dimensional vector function which describes the dynamics, and where G and Q are $m \times p$ and $p \times r$ matrices describe how the uncertainty and controls are distributed within the system; W is Brownian motion (see, e.g. [37] for more detail about stochastic optimal control; I closely follow [21] in the presentation of path integral control).

The "optimal cost-to-go" is defined as

$$J(x,t) = \min_u C(x,t,u),$$

and satisfies the stochastic Hamilton-Jacobi-Bellman (HJB) equation [37]:

$$\begin{aligned} \partial_t J(x,t) &= \partial_x J(x,t) + \mathrm{Tr}\left(Q^T G^T \partial_{xx} J(x,t) G Q\right) + V(x,t) \\ &\quad -\tfrac{1}{2}\partial_x J(x,t) G^T R^{-1} G \partial_x J(x,t), \end{aligned}$$

where Tr is the trace. If there exists a scalar γ such that

$$\gamma G R^{-1} G^T = G Q Q^T G^T, \tag{8}$$

then the nonlinear change of variables

$$J(x,t) = -\gamma \log \psi(x,t),$$

linearizes the stochastic HJB equation and one obtains

$$\partial_t \psi = \frac{V(x,t)}{\gamma}\psi - f(x,t)^T \partial_x \psi - \tfrac{1}{2}\mathrm{Tr}\left(Q^T G^T \partial_{xx}\psi G Q\right), \tag{9}$$

with final condition $\psi(x,t_f) = \exp(-\Phi(x(t_f))/\gamma)$ and with the optimal control

$$u = -R^{-1}\partial_x J\, G, \tag{10}$$

see [12, 21]. Thus, one can compute the optimal control by solving the HJB equation. Numerical PDE solvers typically require a grid on the domain of the PDE, however the domain has the dimension of the state space of the control problem. Thus, grid based numerical PDE techniques only apply to control problems of a low dimension (or else the computational requirements become excessive).

For larger dimensional problems, one can use stochastic PDE solvers which do not require a grid. In particular, one can solve the adjoint equation

$$\partial_t \psi = -\frac{V(x,t)}{\gamma}\psi - \partial_x(f(x,t)\psi) + \tfrac{1}{2}\mathrm{Tr}\left(QG\partial_{xx}\psi G^T Q^T\right),$$

forward in time (instead of solving (9) backwards in time) with the Feynman-Kac formula (see, e.g. [5])

$$\psi(x,t) = E\left[\exp\left(-\frac{1}{\gamma}\left(\Phi(y(t_f),t_f) + \int_t^{t_f} V(y(\tau),\tau)d\tau\right)\right)\right], \tag{11}$$

where the expectation is over the trajectories of

$$dy = f(y,\tau)\,d\tau + GQ\,dW, \tag{12}$$

starting at $y(\tau = t) = x(t)$. This is the path integral formulation of stochastic optimal control [21–23, 38]. Note that the class of problems that can be tackled with path integral control is rather general, since the assumptions of (i) a quadratic control cost; and (ii) the condition on the noise and the control cost in equation (8) are not restrictive. In particular, the dynamics $f(x,t)$ and the potential $V(x,t)$ in (7) can be nonlinear.

To evaluate the Feynman-Kac formula numerically, one approximates the infinite dimensional integral in (11) by a finite dimensional one. For example, one can discretize the integral on a regular grid in time with constant time step Δt, so that

$$\begin{aligned}\psi(x,t) &\approx \int dy_1 \cdots \int dy_n \, p(y_1,\ldots,y_n) \\ &\quad \times \exp\left(-\frac{1}{\gamma}\Phi(y_n, n\Delta t) - \frac{\Delta t}{2\gamma}\sum_{i=1}^{n} V(y_i,\tau_i) + V(y_{i-1},\tau_{i-1})\right), \end{aligned} \tag{13}$$

where $y_1 = x(t)$, $n = (t_f - t)/\Delta t$, $\tau_i = t + i\Delta t$, $i = 0, 1, \ldots, n$, and where p is the pdf of the discretized trajectory $y_1, \ldots, y_n$. Here the trapezoidal rule is used to discretize the integral of the potential, however other choices are also possible [18]. The SDE (12) implies that the increments $\Delta y_i = y_i - y_{i-1}$ are independent Gaussians, e.g. for a forward Euler discretization [24] of (12),

$$\Delta y_i \sim \mathcal{N}\left(f(y_i,\tau_i)\Delta t, \Sigma\Delta t\right),$$

where $\Sigma = \gamma G R^{-1} G^T = GQQ^T G^T$, so that

$$\begin{aligned} p(y_1,\ldots,y_n) &= \prod_{i=1}^{n} p(\Delta y_i) \\ &= \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{n}\Delta t\left(\frac{\Delta y_i}{\Delta t} - f(y_i,\tau_i)\right)^T \Sigma^{-1}\left(\frac{\Delta y_i}{\Delta t} - f(y_i,\tau_i)\right)\right)}{\left(2\pi\Delta t \det\Sigma\right)^{-n/2}}. \end{aligned} \tag{14}$$

Thus,

$$\psi(x,t) \approx \frac{1}{\left(2\pi\Delta t\det\Sigma\right)^{n/2}} \int dy_1 \cdots \int dy_n \exp\left(-F(y_1,\ldots,y_n)\right),$$

where

$$\begin{aligned} F(y_1,\ldots,y_n) &= \frac{1}{\gamma}\Phi(y_n,n\Delta t) + \frac{\Delta t}{2\gamma}\sum_{i=1}^{n} V(y_i,\tau_i) + V(y_{i-1},\tau_{i-1}) \\ &\quad + \frac{1}{2}\sum_{i=1}^{n}\Delta t\left(\frac{\Delta y_i}{\Delta t} - f(y_i,\tau_i)\right)^T \Sigma^{-1}\left(\frac{\Delta y_i}{\Delta t} - f(y_i,\tau_i)\right). \end{aligned} \tag{15}$$

Note that this F depends on how one discretizes the integral of the potential V, and the SDE (12); other discretization schemes will lead to different Fs.

For a given discretization, i.e. for a given F, MC sampling can be applied to compute the expectation in (13). For example, one can evaluate

$$G(y_1,\ldots,y_n) = \frac{1}{\gamma}\Phi(y_n,n\Delta t) + \frac{\Delta t}{2\gamma}\sum_{i=1}^{n} V(y_i,\tau_i) + V(y_{i-1},\tau_{i-1}),$$

for discretized trajectories of (12), followed by averaging. However, this method can fail if the potential V has deep wells [21–23]. In this case, many trajectories of (12) can end up

where V is large and, thus, contribute little to the approximation of the expected value ψ. For efficient sampling, one needs a method that guides the samples to remain where the potential is small.

This guiding of the samples can be achieved via implicit sampling. Recall that in implicit sampling one first locates the region of high probability by minimizing $F = -\log(p(x))$, where $p(x)$ is the pdf of the random variable one is interested in. The trajectories we wish to sample have the joint pdf (14), so that, in order to apply implicit sampling to path integral control, one needs to minimize F in (15). With this F, one solves the algebraic equations (1) with a one-to-one and onto map (2). The integral (13) becomes

$$\begin{aligned} \psi(x,t) &\approx \frac{\exp(-\phi)}{(2\pi\Delta t \det\Sigma)^{n/2}} \int d\xi_1 \cdots \int d\xi_n \exp(-\xi^T\xi/2) \left|\frac{\partial x}{\partial \xi}\right| \\ &\approx \frac{\exp(-\phi)}{(\Delta t \det\Sigma)^{n/2}} E_\xi\left[\left|\frac{\partial x}{\partial \xi}\right|\right], \end{aligned} \tag{16}$$

where the expectation is over the reference variable ξ and where $\partial x/\partial\xi$ is the Jacobian of the map from x to ξ in equation (2). Combining (16) with the expression (3) for the Jacobian gives

$$\psi(x,t) \approx \frac{\exp(-\phi)}{2\,(\Delta t \det\Sigma)^{n/2} \det(L)} E_\xi\left[(\xi^T\xi)^{(1-m)/2}\left|\frac{\lambda(\xi)^{m-1}}{\nabla F(\xi)\,\eta}\right|\right].$$

The expected value is now straightforward to compute via Monte Carlo, i.e. sampling a the reference variable ξ to obtain M samples ξ^j, $j = 1,\ldots,M$, computing, for each one, $\lambda(\xi^j)$ and the gradient of F, followed by averaging. In numerical implementations, it may be more efficient to use the equivalent expression (3) for the Jacobian (which avoids computing the gradient of F). In this case, one obtains

$$\psi(x(t),t) \approx \frac{\exp(-\phi)}{(\Delta t \det\Sigma)^{n/2} \det(L)} E_\xi\left[(\xi^T\xi)^{(1-m)/2}\left|\lambda(\xi)^{m-1}\frac{d\lambda}{d\rho}(\xi)\right|\right].$$

Once $\psi(x,t)$ is computed, we can compute the optimal control from (10), e.g. via finite differences. It is important to note that this defines the optimal control at time t, and that the computations have to be repeated at the next time step to compute $\psi(x(t+dt),t+dt)$. The use of implicit sampling in stochastic optimal control is illustrated with three examples.

3.1 The double slit problem

The double slit problem is a simple example that illustrates the pitfalls one must avoid when using MC sampling for path integral control [21]. The problem is as follows. Suppose you observe a somehow confused person walking (randomly) towards a wall with two doors, and your job is to guide this person through either one of the two doors. The person and you are modeled by the controlled dynamics

$$dx = u\,dt + \sqrt{\sigma}\,dW,$$

Table 1: Parameters of the double slit problem

Description	Parameter	Value
Final time	t_f	2
Critical time	t_1	1
Slits	a, b, c, d	$-6, -4, 6, 8$
Initial position	$x(0)$	1
Variance of the noise	σ	1
Control cost	r	0.1
Time step	Δt	0.02

the final cost is quadratic, $\Phi = x(t_f)^2/2$, and the scalar $R > 0$, that defines the cost of the control, is given; the "wall" is modeled by the potential

$$V(x,t) = \begin{cases} \infty & \text{if } t = t_1 \text{ and } x < a, \text{ or } b < x < c, \text{ or } d < x \\ 0 & \text{otherwise,} \end{cases}$$

for given $a < b < c < d$ and $t_1 > 0$. The stochastic HJB equation of the double slit problem is

$$\psi_t = -\frac{Q}{\gamma} + \frac{1}{2}\sigma\psi_{xx},$$

where $\gamma = r\sqrt{\sigma}$ satisfies (8).

One can attempt to solve this equation using standard MC sampling as follows. Solve the SDE

$$dy = \sqrt{\sigma}\, dW,$$

starting at $y(\tau = t) = x(t)$ repeatedly, e.g. using a uniform grid in time and the forward Euler scheme [24]

$$y_{n+1} = y_n + \sqrt{\sigma\Delta t}\Delta w^n,$$

where Δw^n are independent Gaussians with mean 0 and variance 1. For each trajectory $\{y_1, \ldots, y_n\}$, evaluate

$$G(y_1, \ldots, y_n) = \frac{1}{2\gamma}y_n^2 + \frac{\Delta t}{2\gamma}\sum_{i=1}^{n} V(y_i, \tau_i) + V(y_{i-1}, \tau_{i-1}).$$

With a high probability, the trajectories will hit the potential wall at t_1 and, thus, G is infinite, so that the contribution of a trajectory to the expected value $\psi(x,t)$ in(11) is zero with a high probability. The situation is illustrated with a simulation using the parameters of table 1. The results are shown in the left panel of figure 1. One observes that 5000 unguided random walks from (17), all starting at $x(0)$, hit the potential wall, and, thus, score $G = \infty$. All 5000 samples thus carry no probability and do not contribute to the approximation of ψ.

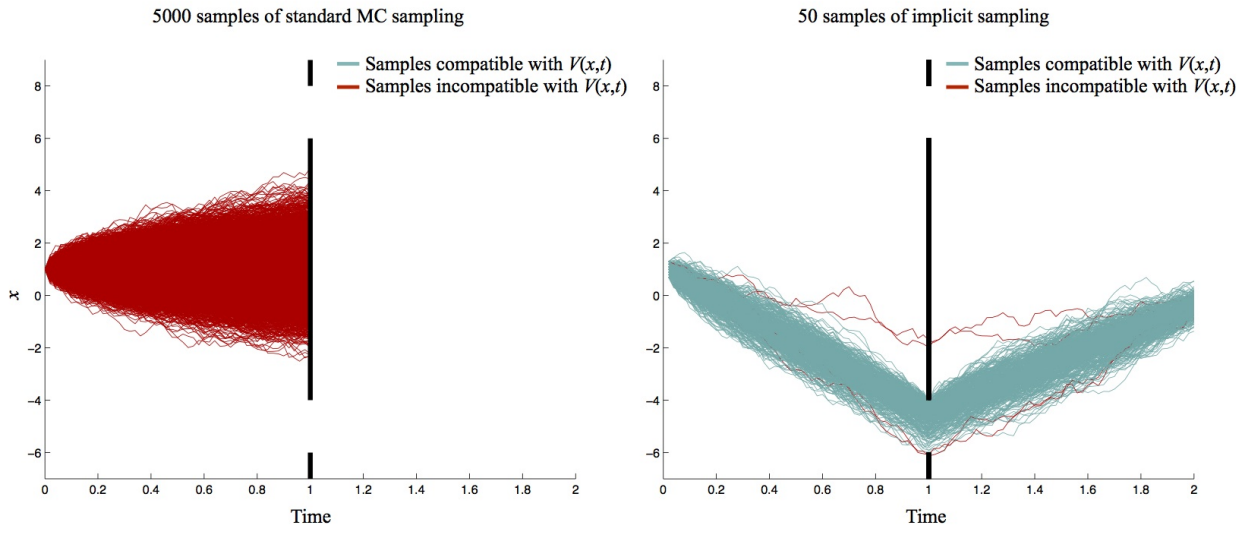

Figure 1: Random walks for the double slit problem. Left: standard MC sampling with 5000 unguided random walks, all of which hit the potential wall. Right: 50 guided random walks obtained via implicit sampling, only 4 of which hit the potential wall.

Even when $O(10^5)$ samples are considered, it is unlikely that sufficiently many make it past the potential wall [21]. We conclude that this method is unfeasible for this problem, since the number of samples required is extremely large due to the deep wells in the potential.

Implicit sampling can be applied here to fix these problems. In particular, one finds from the Feynman-Kac formula that

$$\psi(x,t) = E\left[\exp\left(-\frac{y(t_f)^2}{2\gamma} - \frac{1}{\gamma}\int_t^{t_f} V(y(\tau),\tau)d\tau\right)\right],$$

which one can compute with implicit sampling in two parts. For $t \geq t_1$, i.e. after one has passed the potential wall, the probability of each path is Gaussian with variance $s = t_f - t$, so that

$$\psi(x,t) = \int \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{(z-y)^2}{2s}\right) \exp\left(-\frac{z^2}{2\gamma}\right) dz \quad \text{for } t \geq t_1.$$

For implicit sampling, one defines

$$F(y) = \frac{(z-y)^2}{2s} + \frac{z^2}{2\gamma},$$

whose minimizer and minimum are $\mu = \text{argmin } F = y\gamma/(s+\gamma)$, $\phi = \min F = y^2/(s+\gamma)$. Since F is quadratic, $F = \phi + H(y-\mu^2)/2$, where $H = (s+\gamma)/(s\gamma)$ is the Hessian of F at the minimum. The algebraic equation (1) can thus be solved by the linear map

$$y = \mu + \sqrt{\frac{s\gamma}{s+\gamma}}\xi,$$

so that

$$\psi(x,t) = \exp(-\phi)\sqrt{\frac{\gamma}{(s+\gamma)}}, \quad \text{for } t \geq t_1.$$

For $t < t_1$, one splits the integration into two parts, first going from time t to t_1 with probability $p(y_1, t_1|y, t)$, and then from t_1 onwards to t_f with probability $p(z, t_f|y_1) = \mathcal{N}(y_1, s)$, $s = t_f - t_1$:

$$\psi(x,t) = \int dz \int dy_1 \exp\big(-\frac{z^2}{2\gamma}\big) p(z, t_f|y_1) p(y_1, t_1|y, t).$$

Implicit sampling uses the information about the potential which is infinite at t_1 except for the two slits, so that $p(y_1, t_1|y, t) = 0$ outside the slits and Gaussian with mean y and variance $\hat{s} = t_1 - t$ otherwise. Since only the slits need to be explored with samples, the integration can be carried out over the slits:

$$\psi(x,t) = \int dy \int_{y\in[a,b]\times[c,d]} dy_1 \exp\big(-\frac{z^2}{2\gamma}\big) \exp\big(-\frac{(z-y_1)^2}{2s}\big) \exp\big(-\frac{(y_1-y)^2}{2\hat{s}}\big).$$

The evaluation of the above integral using the same strategy as above for $t \geq t_1$ is tedious, but straightforward. Note that the implicit sampling strategy here is the key to solving this problem, because it locates the wells in the potential.

The above analytic solution is compared to a numerical implementation of implicit sampling, for which the paths are discretized with a constant time step Δt. Here the numerical integration is also split up into two parts. For $t \geq t_1$ the function F is quadratic because the probabilities are Gaussian and the potential has no effect. Thus,

$$F = \frac{y_n^2}{2\gamma} + \frac{1}{2\gamma\Delta t}\sum_{i=1}^{n}(y_i - y_{i-1})^2,$$

where $n = (t_f - t)/\Delta t$ and $y_i = y(t + i\Delta t)$,$i = 1, \ldots, n$. Minimizing F is straight forward, and the algebraic equation (1) can be solved with a linear map

$$y = \mu + L^{-T}\xi,$$

where $y = (y_0, \ldots, y_n)$ is a n-dimensional vector, μ is the minimizer of F and ξ is an n-dimensional vector whose elements are independent standard normal variates; L is a Cholesky factor of the Hessian at the minimum. Since the Jacobian of this linear map is constant, (16) gives

$$\psi(x,t) = \frac{\exp(-\phi)}{(\gamma\Delta t)^{n/2}\det L},$$

where ϕ is the minimum of F.

For $t < t_1$, one obtains the same F, but needs to perform a constraint minimization over the slits. There are two (local) minima, one per slit, and both can be found easily using quadratic programming [13,34]. One can then generate a sample close to each of the minima using

$$y = \mu + L^{-T}\xi_j,$$

where y again is a vector whose elements are the discretized path and where μ is the location of a local minimum of the constraint problem and L is a Cholesky factor of the unconstrained Hessian at a minimum. Equation (16) becomes

$$\psi(x,t) \approx \frac{\exp(-\phi)}{(\Delta t\gamma)^{n/2}} \int_{\text{Slits}} \frac{1}{\det L} d\xi.$$

The expected value of the Jacobian $(1/\det L)$ over the slits can be computed by Monte Carlo as follows. Generate M samples ξ^j, $j = 1, \ldots, M$, and, for each one, compute the corresponding trajectory and check if it hits the potential wall. The integral is $1/\det(L)$ times the ratio of the number of trajectories that pass through the wall and M.

The right panel of figure 1 shows 50 trajectories one obtains with this approximation at $t = \Delta t$. Note that most of the trajectories pass through the slit, i.e. most of the samples carry a significant probability, score a small F, and, thus, contribute to the approximation of the expected value $\psi(x,t)$. These "guiding" effects make it possible to solve the problem with $O(10)$ samples, while the standard Monte Carlo scheme fails even with $O(10^5)$ samples [21]. The Laplace guided strategy presented in [21] (which uses similar ingredients as implicit sampling and is also related to the optimal nudging constructions in [43, 44]) requires about 100 samples.

The numerical approximation obtained with implicit sampling (50 samples) is compared to the analytical solution in figure 2, where the optimal control and the trajectory under this control are shown. One observes that the numerical approximation of the optimal control is quite close to the control one obtains from the analytical solution (right panel of figure 2) and, hence, the controlled trajectory one obtains with implicit sampling is also close to the one computed from the analytical solution (left panel of figure 2). This statement can be made more precise by solving the control problem repeatedly (each solution is a random event) and averaging.

The double slit problem is solved 500 times and the error of the numerical approximation is recorded for each run. The Euclidean norm of the difference of the analytical solution and the approximation via implicit sampling is used to measure an error, in particular in x (the trajectory under optimal control) and u (the optimal control). The number of samples of implicit sampling is varied to study the convergence of the algorithm. The results are shown in table 2 where the mean and standard deviation of the Euclidean norm of the error in x, respectively u, scaled by the mean of the norm of x and u respectively, are listed. These numbers indicate the errors one should expect in a typical run. The errors are relatively small when 50 samples are used. The errors do not dramatically decrease for larger sample numbers, which indicates that the algorithm has converged. The small variance of the errors indicates that similar errors occur in each run, so that one concludes that implicit sampling is reliable.

The error in the numerical implementation of implicit sampling is mostly due to neglecting one of the local minima of F. In numerical tests, only a slight improvement was observed when both minima were used for sampling, however the computations are twice as fast if one considers only the smaller of the two minima.

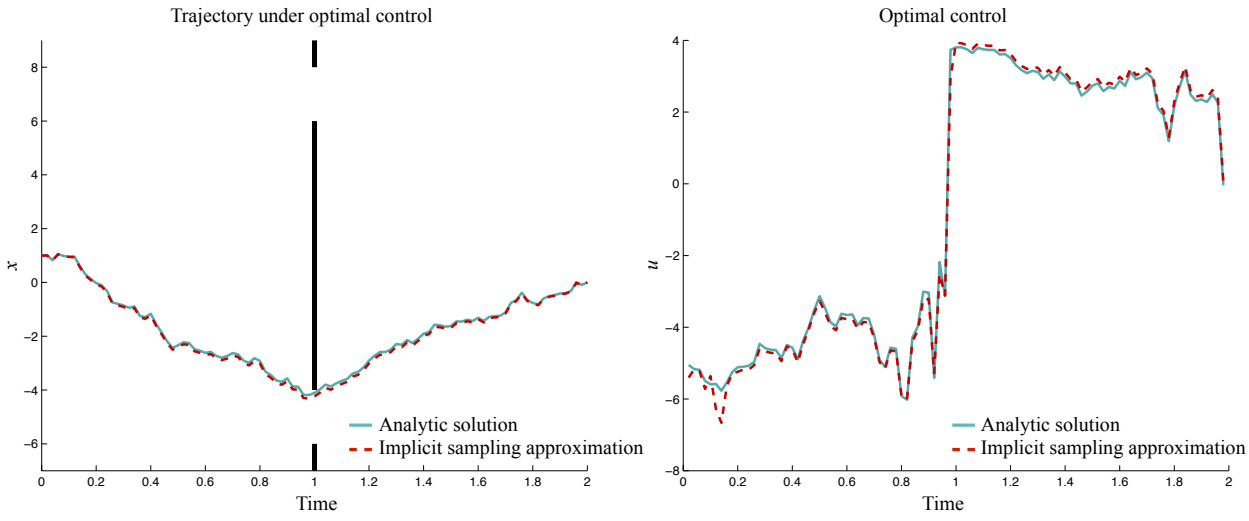

Figure 2: Comparison of the analytical solution and its approximation by implicit sampling with 50 samples. Left: the trajectory under optimal control (solid turquoise) and its numerical approximation (dashed red). Right: the optimal control (solid turquoise) and its numerical approximation (dashed red).

3.2 Stochastic control of a two-degrees-of-freedom robotic arm

Stochastic optimal control of the two-degrees-of-freedom robotic arm shown in figure 3 is considered. The controller is an "inverse dynamics controller" that linearizes the system. This example demonstrates how implicit sampling based path integral control works in linear problems. Specifically, a semi-analytical solution is derived for which a numerical optimization is required, however the expected value of the implicit sampling solution in (16) can be evaluated explicitly (i.e. no numerical sampling is needed). Moreover, only some of the dynamic equations of the first order formulation of the dynamics are driven by noise, so that the dimension of the stochastic subspace is less than the actual dimension of the problem. Implicit sampling can exploit this features, and this example demonstrates how. The algo-

Table 2: Mean errors of the numerical implementation of implicit sampling

	Number of samples			
	10	50	100	500
Error in x in % (mean±std. dev.)	2.64±0.63	2.70±0.63	2.66±0.63	2.61±0.62
Error in u in % (mean±std. dev.)	4.81±2.15	4.61±1.89	4.54±2.37	4.46±2.17

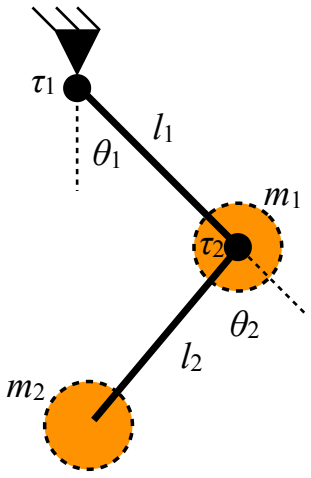

Figure 3: Sketch of a two-degrees-of-freedom robotic arm.

rithm is tested in numerical experiments in which false parameters are given to the algorithm to demonstrate the robustness of the path integral controller to model error.

The goal is to compute two independent torques τ_1 and τ_2 that drive the arm to a desired position θ^* (described by the two-angles θ_1, θ_2, the degrees-of-freedom). Neglecting energy dissipation, and assuming the robot is mounted on a horizontal table (no effects of gravity), the equation of motion is

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} = \tau$$

where $\theta = (\theta_1, \theta_2)$ is a 2-dimensional vector and where dots denote derivatives with respect to time; the 2×2 matrices

$$\begin{aligned} M(\theta) &= \begin{pmatrix} a_1 + a_3\cos(\theta_2) & a_2 + a_3\cos(\theta_2) \\ a_2 + a_3\cos(\theta_2) & a_2 \end{pmatrix}, \\ C(\theta, \dot{\theta}) &= \begin{pmatrix} -a_3\sin(\theta_2)\dot{\theta}_2 & -a_3(\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_2) \\ a_3\sin(\theta_2)\dot{\theta}_1 & 0 \end{pmatrix}, \end{aligned}$$

where

$$\begin{aligned} a_1 &= m_1 l_{c1}^2 + m_2 l_{c1}^2 + m_2 l_{c2}^2 + I_1 + I_2 \\ a_2 &= m_2 l_{c2}^2 + I_2 \\ a_3 &= l_1 m_2 l_{c2} \end{aligned}$$

define the dynamics and depend on the lengths of the arms l_1, l_2 and the loads m_1, m_2 (see, e.g. [35] for more detail about this model and table 3 for the parameters used in simulations). One can compute the torques by inverting the dynamics and choosing $\tau = C(\theta, \dot{\theta}) + M(\theta)u$ to derive the linear system

$$\begin{pmatrix} \dot{\theta} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} + \begin{pmatrix} 0 \\ I \end{pmatrix} u, \tag{17}$$

Table 3: Parameters of the robotic arm

Description	Parameter	Value
Length of link one	l_1	1 m
Distance to the center of mass of link one	l_{c1}	0.5 m
Distance to the center of mass of link two	l_{c2}	0.5 m
Mass of link one	m_1	1 kg
Mass of link two	m_2	1 kg
Moment of inertia of link one	I_1	2 kg m^2
Moment of inertia of link two	I_2	2 kg m^2

where 0 denotes the matrix whose elements are all zero (of appropriate dimensions), and I is the 2×2 identity matrix; u is a 2-dimensional vector of controls which must be computed. To make the control robust to modeling errors, one can add noise and solve the stochastic optimal control problem

$$dx = A\,dt + Gu\,dt + GQ\,dW,$$

where $x = (\theta, \dot{\theta})$ is a 4-dimensional vector and where the matrices A, G and Q can be read from (17), and where W is a 2-dimensional Brownian motion. The final cost is chosen as

$$\Phi = ||\theta(t_f) - \theta^*||^2/2 + ||\dot{\theta}(t_f)||^2/2,$$

so that the robotic arm stops at $t_f > 0$ at the desired position θ^*. This linear problem can be solved with linear quadratic Gaussian (LQG) control [37], however here a semi-analytical solution is obtained via implicit sampling.

As before, the time is discretized using a constant time step $\Delta t = (t_f - t)/n$. The discretized dynamics are

$$\begin{aligned} z_{i+1} &= z_i + y_i\,\Delta t, \\ y_{i+1} &= y_i + \Delta B, \end{aligned}$$

where y_i, z_i are 2-dimensional vectors whose elements are the discretized angular velocities ($\dot{\theta}$) and the discretized angles (θ); note that only one of the above equations is driven by noise (ΔB), since there is no reason to inject noise into the (trivial) equation $\dot{\theta} = \dot{\theta}$. While the noise propagates via the coupling to all variables, the pdfs that define the path in (11) are in terms of y_ns only.

The function F in implicit sampling is thus a function of y only,

$$F = \frac{1}{2\gamma} y_n^2 + \frac{1}{2\gamma}(z_n - \theta^*)^2 + \frac{1}{2\Delta t}\sum_{i=1}^{n}(y_i - y_{i-1})^2,$$

where $\gamma = r$ is chosen to satisfy (8). Because F is quadratic the minimization is straightforward, and the algebraic equation (1) can be solved with a linear map whose Jacobian is

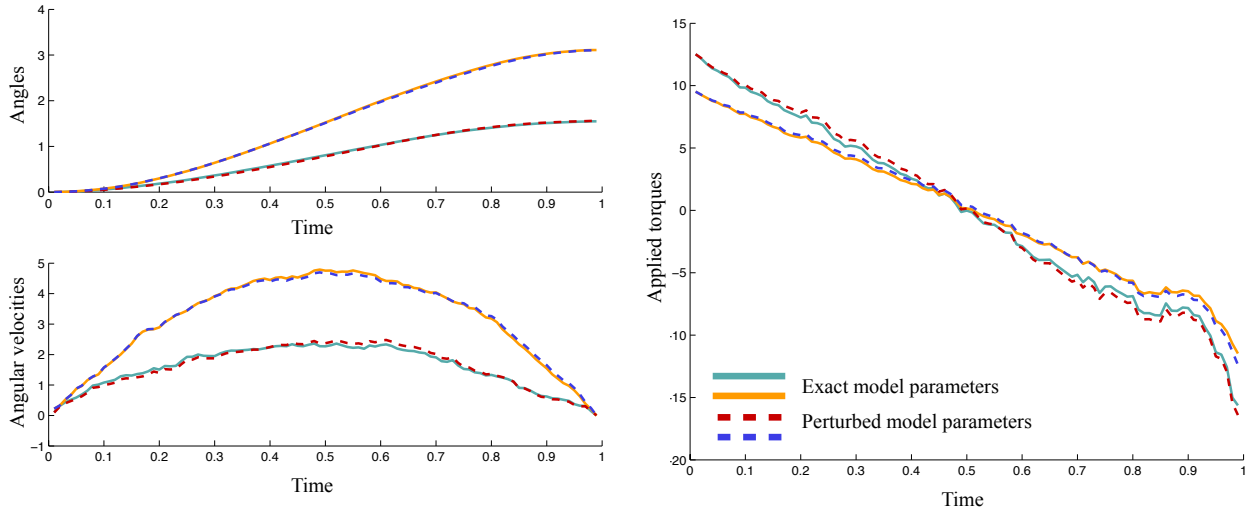

Figure 4: Simulation of a robotic arm under path integral control. Upper left: trajectories of angles θ_1 (turquoise/red) and θ_2 (orange/blue); lower left: trajectories of angular velocities $\dot{\theta}_2$ (turquoise/red) and $\dot{\theta}_2$ (orange/blue); right: applied torques τ_1 (turquoise/red)and τ_2 (orange/blue). Solid turquoise and orange lines: trajectories and torques under optimal control with exact model parameters; dashed red and blue lines: trajectories and torques under path integral control with false model parameters.

constant and given by the determinant of a Cholesky factor L of the Hessian of F at the minimum (see above). Thus, equation (16) becomes

$$\psi(x,t) = \frac{\exp(-\phi)}{(\Delta t)^n \det L}.$$

There is no need for generating samples, since the expected value is computed explicitly (i.e. by evaluating the integral analytically).

The robustness of the implicit sampling based path integral control is tested in numerical experiments. To simulate that the "real" robotic arm behaves differently from the numerical model that the path integral controller uses to find a control, one can give the controller false information about the parameters of the numerical model. Here the parameters m_1 and m_2 are perturbed values of the true parameters simulating that the robotic arm picked up a payload (of unknown weight). Thus, the controller works with values $m_1 = 1.4$, $m_2 = 1$, whereas the "true" robotic arm has parameters as in table 3. The results of a simulation are shown in figure 4 where state trajectories and controls are shown for a truly optimal controller (i.e. one who has access to the exact model parameters), and for the path integral controller which uses false model parameters. One observes that the path integral controller can perform the desired task (moving the arm to a new position) and that its control and the resulting state trajectories closely follow those that are generated by a truly optimal

controller. This example thus indicates that the path integral controller can perform reliably and quickly in applications where model error may be an issue.

3.3 Optimization via stochastic control

One can set up a conventional optimization problem, i.e. find $\min f(x)$ for a given smooth function f, as a stochastic optimal control problem as follows: find a control u to minimize the cost function

$$C(x, t_f) = E\left[f(x(t_f)) + \int_0^{t_f} u(x,t)^T R u(x,t) dt\right], \tag{18}$$

where the expectation is over trajectories of the SDE

$$dx = u\, dt + \sqrt{\sigma}\, dW.$$

The idea is to make use of the stochastic component to explore valleys in the function f. The parameters one can tune are the noise level σ, the final time t_f and the control cost R.

Implicit sampling for this control problem requires minimization of the function

$$F = \frac{1}{2\sigma\Delta t}\sum_{i=1}^{n}(y_i - y_{i-1})^2 + \frac{1}{\gamma} f(y_n),$$

where the discretization is done using a constant time step $\Delta t = t_f/n$ as before. Note that the control approach to this problem inserts a quadratic term through which the space is (randomly) explored.

These ideas are applied to minimize the Himmelblau function

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2,$$

which is a popular test of the performance of optimization algorithms because of its many local minima [19]. The parameters are $R = 0.01, \sigma = 0.01, \Delta t = 1, t_f = 20$, and the minimization is initialized at $(-1, -4)$. Figure 5 shows the iterations obtained with implicit sampling and 50 samples, and, for comparison, the iterations of Newton's method.

In this example, the stochastic approach is successful and finds a much lower minimum than Newton's method. However, since each run of the stochastic approach is random, one may find another local minimum in another run. This could be useful when one needs to explore valleys or if one suspects the existence of other local minima.

To test the reliability of the stochastic control approach, 70 experiments were performed, each starting at the same initial condition. The iterations of 5 such experiments are shown in table 4. All 70 runs ended up close to the local minimum around $x_1 = 2.5$, $x_2 = -2.5$. The average value of F is an approximation to C in (18) and, with 70 experiments was found to be $C \approx 1.75$, which corresponds to a much lower value of f than the value $f_{\min}^{\text{Newton}} = 178.34$, that one obtains with Newton's method.

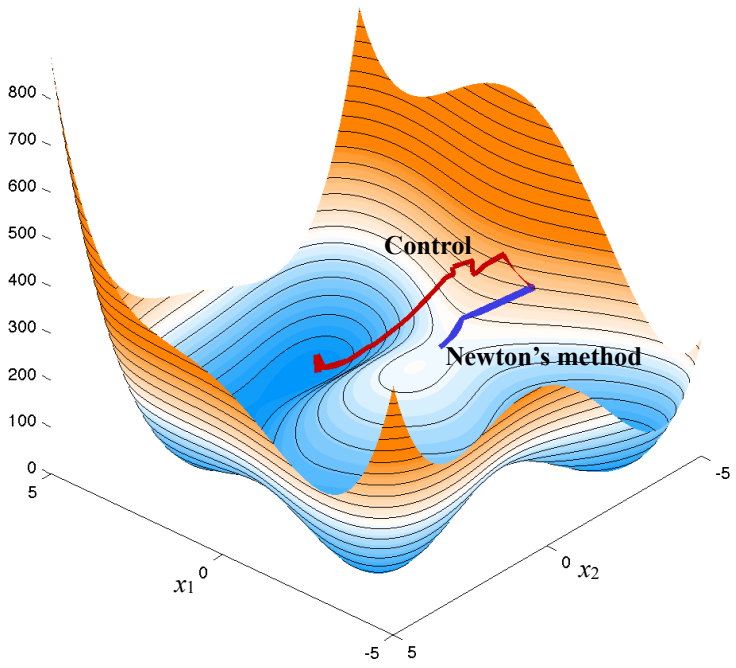

Figure 5: Newton's method and stochastic optimization for the Himmelblau function.

4 Application to Monte Carlo localization and SLAM

Consider a mobile robot that navigates autonomously and, as it moves, collects noisy measurements about its motion as well as scans of its environment. If the scans reveal locations of features that are known to the robot, i.e. if the robot has a map of its environment, then it can localize itself on this map. This problem is known as localization. When the robot has no map of its environment, then it must construct a map while localizing itself on it, leading to the problem of simultaneous localization and mapping (SLAM) [9, 25, 39]. Efficient solutions to the localization and SLAM problems are a fundamental requirement for autonomous robots, which must move through unknown environments where no global positioning data are available, for example indoors, in abandoned mines, underwater, or on far-away planets [15,40,41]. Application of implicit sampling to localization and SLAM is the subject of the next two sections, where the algorithms will also be tested using experimental data obtained by Nebot and colleagues at the University of Sydney [32].

4.1 Monte Carlo localization

The localization problem can be formulated as follows. Information about the initial state of the robot is available in the form of a pdf $p(x_0)$, where x_0 is an m-dimensional vector whose elements are the state variables (e.g. position and velocity of the robot). A probabilistic

Table 4: Newton's method and 5 runs of controlled optimization

Iteration	Newton	Controlled Optimization				
0	260.00	260.00	260.00	260.00	260.00	260.00
1	195.36	268.57	266.47	241.73	337.77	257.55
2	180.19	258.45	242.92	239.19	347.24	263.20
3	178.43	259.75	268.95	258.97	360.16	243.92
4	178.34	257.44	279.41	249.28	373.97	218.95
5	178.34	214.25	243.23	224.21	336.47	201.69
6	178.34	184.70	320.29	256.08	476.23	179.34
7	178.34	161.31	331.32	230.16	394.82	157.00
8	-	147.39	330.16	218.89	316.97	150.73
9	-	144.04	255.65	175.61	245.24	150.33
10	-	101.05	221.05	180.92	176.98	144.97
11	-	76.729	188.49	174.13	180.53	105.48
12	-	48.46	151.54	188.66	131.86	91.42
13	-	33.05	115.34	171.75	101.82	67.00
14	-	29.90	87.37	147.43	56.51	41.29
15	-	41.34	62.89	104.69	35.11	10.65
16	-	31.03	43.69	72.71	31.48	6.08
17	-	18.12	22.62	42.44	14.53	1.79
18	-	1.83	9.195	30.55	9.52	2.61
19	-	0.46	1.90	0.54	0.97	1.24

motion model defines the pdf

$$p(x_n|x_{n-1}, u_n), \tag{19}$$

where $n = 1, 2, \ldots$ is discrete time and where u_n is a p-dimensional vector of known "controls", e.g. the odometry. A data equation describes the robot's sensors and defines the pdf

$$p(z_n|x_n, u_n, \Theta), \tag{20}$$

where z_n is a k-dimensional vector whose elements are the data ($k \leq m$) and where Θ is the map. For example, one can use the "landmark model", for which the map describes the coordinates of a collection of obstacles, called"landmarks"; the data are measurements of range and bearing of the position of the robot relative to a landmark. The landmark model and range and bearing data will be used in the applications below, but the algorithms described are more generally applicable.

The motion and sensor model jointly define the conditional pdf $p(x_{0:n}|z_{0:n}, u_{0:n}, \Theta)$, where the short-hand notation $x_{0:n} = \{x_0, x_1, \ldots, x_n\}$ is used to denote a sequence of vectors. By

using Bayes' rule repeatedly, one can derive the recursive formula

$$p(x_{0:n}|z_{0:n}, u_{0:n}, \Theta) = p(x_{0:n-1}|z_{0:n-1}, u_{0:n-1}, \Theta)\frac{p(x_n|x_{n-1}, u_n)p(z_n|x_n, u_n, \Theta)}{p(z_n|z_{0:n-1})}.$$

Approximations of this pdf are used to localize the robot. For example, methods based on the Kalman filter [20] (KF) construct a Gaussian approximation which is often problematic because of nonlinearities in the model and data. Moreover, KF-based localization algorithms can diverge, for example, in multi-modal scenarios (i.e. if the data are ambiguous) or during re-localization after system failure [26]. The basic idea of Monte Carlo localization (MCL) is to relax the Gaussian assumptions required by KF, and to apply importance sampling to the localization problem [8]. The method proceeds as follows.

Given M weighted samples $\{X^j_{0:n-1}, W^j_{n-1}\}$, $j = 1, \ldots, M$, which form an empirical estimate of the pdf $p(x_{0:n-1}|z_{0:n-1}, u_{0:n-1}, \Theta)$ at time $n-1$, one updates each particle to time t_n given the datum z_n. In standard MCL [8], this is done by choosing the importance function

$$(p_0)_n \propto p(x_{0:n-1}|z_{0:n-1}, u_{0:n-1}, \Theta)p(x_n|X^j_{n-1}, u_n),$$

i.e. the robot location is predicted with the motion model and then this prediction, X^j_n, is weighted by

$$W^j_n \propto W^j_{n-1}p(z_n|X^j_n, u_n, \Theta),$$

to assess how well it fits the data. However, this choice of importance function can be inefficient, especially if the motion model is noisy and the data are accurate (as is often the case [40]). The reason is that many of the samples generated by the model are unlikely with respect to the data and the computations used to generate these samples are wasted.

Implicit sampling can be used to speed up the computations. This requires in particular that implicit sampling, as described in section 2, is applied to the M functions

$$F^j(x_n) = -\log\left(p(x_n|X^j_{n-1}, u_n)p(z_n|x_n, u_n, \Theta)\right). \tag{21}$$

One needs M functions F^j, $j = 1, \ldots, M$, one per sample, because the recursive problem formulation requires a factorization of the importance function (compare to sections 2 and 3, where only one function was needed). Here each function F_j is parametrized by the location of the sample at time $n-1$, X^j_{n-1}, the control u_n and the datum z_n; the variables of these functions are the location of the robot at t_n, x_n.

For MCL with implicit sampling, each function F^j must be minimized, e.g. using Newton's method. After the minimization, one can generate samples by solving the stochastic equation (1) with the techniques described in section 2. Using information from derivatives in sampling for MCL has also been considered in [3, 4].

4.1.1 Implementation for the University Car Park data set

The University Car Park data set [32] is used to demonstrate the efficiency of MCL with implicit sampling. The scenario is as follows. A robot is moving around a parking lot and

steering and speed data are collected via a wheel encoder on the rear left wheel and a velocity encoder. An outdoor laser (SICK LMS 221) is mounted on the front bull-bar and is directed forward and horizontal (see [33] for more information on the hardware). The laser returns measurements of relative range and bearing to different features. Speed and steering data are the controls of a kinematic model of the robot and the laser data are used to update this model's predictions about the state of the robot.

The motion model is the forward Euler discretization of the continuous time 2D-model in [33],

$$x_{n+1} = x_n + R(x_n, u_n)\delta + \Delta B_n, \quad \Delta B_n = \mathcal{N}(0, (\Sigma_1)_n),$$

where $R(x_m, u_n)$ is a 3-dimensional vector function, δ is the time step, and $(\Sigma_1)_n$ is a given covariance matrix. The data equation is

$$z_n = h(x_n, \Theta) + V_n, \quad V_n \sim \mathcal{N}(0, \Sigma_2),$$

where h is a 2-dimensional vector function that maps the position of the robot to relative range and bearing and where Σ_2 is a 2×2 diagonal matrix (see [33] and the appendix for the various model parameters).

With the above equations for motion model and data, the functions F^j of implicit sampling in (21) become

$$F^j(x_n) = (x_n - f_n^j)^T \left((\Sigma_1)_n\right)^{-1} (x_n - f_n^j) + \sum_{i=1}^{p} (h(x_n) - z_n^i)^T \Sigma_2^{-1} (h(x_n) - z_n^i), \tag{22}$$

where $f_n^j = X_{n-1}^j + R(X_{n-1}^j, u_n)\delta$ and where z_n^i denotes the measurement for the ith landmark at time n. These F^js are minimized with Newton's method and samples are generated by solving a quadratic equation as explained in section 2. The gradient and Hessian in Newton's method were computed analytically (see appendix). If uneven weights were observed, a "resampling" was done using Algorithm 2 in [1]. Resampling replaces samples with a small weight with samples with a larger weight without introducing significant bias. If no laser data are available, all samples evolve according to the model equation (19).

The results of MCL with implicit sampling are shown in the left panel of Figure 6. The "ground truth" shown here is the result of an MCL run with GPS data (these GPS data however are not used in implicit sampling); the large blue dots are the positions of the landmarks. The reconstruction by the MCL algorithm (dashed turquoise line) is an approximation of the conditional mean, which is obtained by averaging the samples. One observes that MCL with implicit sampling gives accurate estimates whenever laser data are available. After periods during which no laser data were available, one can observe a strong "pull" towards the true state trajectory, as soon as data become available, as is indicated by the jumps in the trajectory e.g. around $x = 0$, $y = 0$.

The efficiency of MCL with implicit sampling is studied by running the algorithm with 10, 20, 40, 80, 100 and 150 samples, i.e. with increasing computational cost, and comparing the reconstruction of the MCL with the true path. The error is measured by the Euclidean

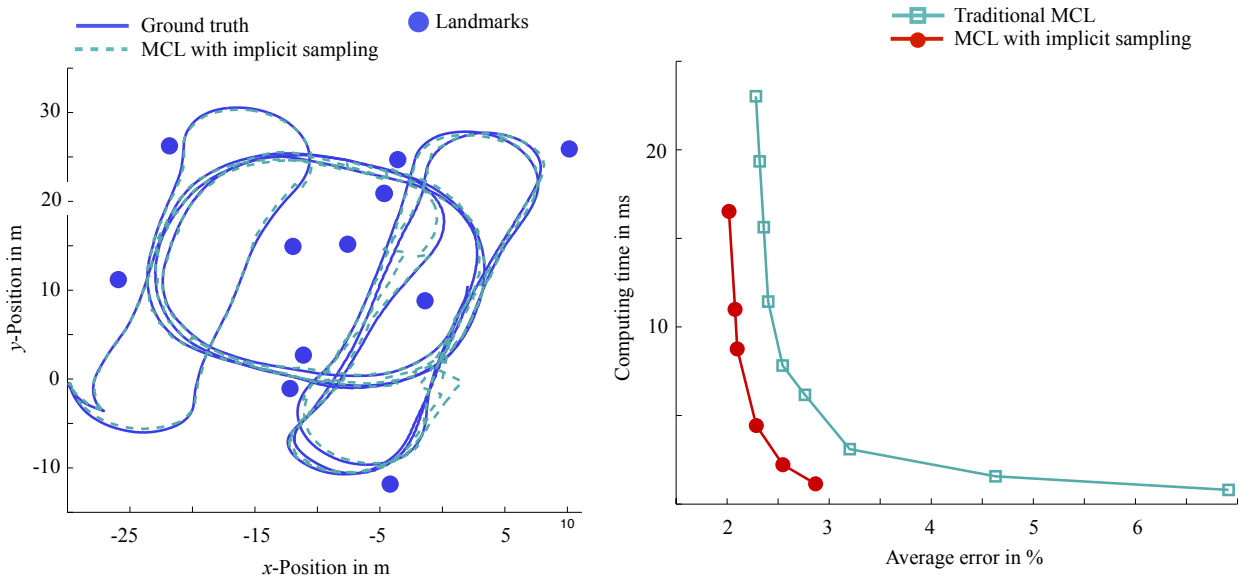

Figure 6: Simulation results of MCL with implicit sampling and standard MCL. Left: the path of the robot and its reconstruction via MCL with implicit sampling (100 samples). Right: average computing time as a function of the average error for standard MCL and MCL with implicit sampling.

norm of the difference between the "ground truth" (see above) and the reconstruction by the MCL algorithm (scaled by the Euclidean norm of the ground truth). The standard MCL method was also applied with 10, 20, 40, 80, 100, 150, 250 and 300 samples. The results are shown in table 5.

It is clear that the both MCL algorithms converge to the same error, since both algorithms converge to the conditional mean as the number of samples (and hence, the computational cost) goes to infinity. However the convergence rate of MCL with implicit sampling is faster, which can be seen from the right panel of figure 6, where the computing time is plotted as a function of the error. In this figure, the area between the lines corresponds to the improvement of implicit sampling over the standard method, and it is clear that implicit sampling is more efficient than the standard method. The improvement is particularly pronounced if a high accuracy is required, in which case MCL with implicit sampling can be orders of magnitudes faster than the standard method.

4.2 Implicit sampling for online SLAM

In SLAM, one is given the probabilistic motion model (19) and a data equation (20), and one needs to estimate the position of the robot as well as the configuration and geometry of

Table 5: Simulation results for MCL

	Implicit sampling		Standard sampling	
# of samples	Error in %	CPU time in ms	Error in %	CUP time in ms
10	2.87	1.14	6.91	0.81
20	2.55	2.23	4.63	1.57
40	2.29	4.42	3.20	3.12
80	2.10	8.80	2.76	6.20
100	2.08	11.0	2.54	7.75
150	2.02	16.5	2.40	11.6
200	-	-	2.36	15.8
250	-	-	2.32	19.6
300	-	-	2.28	23.4

the map; the conditional pdf of interest is thus

$$p(x_{0:n}, \Theta | z_{0:n}, u_{0:n}, \eta_{0:n}), \tag{23}$$

where the map, Θ, is a variable (not a given parameter as in MCL); the variable $\eta_{0:n}$ is the "data association", i.e. it maps the data to the known features, or creates a new feature if the data are incompatible with current features [11, 36]. Here it is assumed that the data association is done by another algorithm, so that $\eta_{0:n}$ in (23) can be assumed to be given (in the numerical implementation in section 4.2.1, a maximum-likelihood algorithm is used [26]).

Note that the SLAM state-vector is different from the state of the localization problem: it is the number of variables needed to describe the robot's position, x, and the coordinates of the features of the map, Θ. If the map contains many features (which is typically the case), then the state dimension is large and this makes KF based SLAM prohibitively expensive. The reason is that KF SLAM requires dense matrix operations on matrices of size N (where N is the number of features), due to correlations between the robot's position and the features [36]. KF SLAM thus has N^2 memory requirements which make it infeasible for realistic N.

Monte Carlo approaches to SLAM reduce the computational cost by exploring conditional independencies [42]. In particular, the various features of the map conditioned on the robot path are independent, which implies the factorization

$$p(\Theta | x_{0:n}, z_{0:n}, u_{0:n}, \eta_{0:n}) = \prod_{k=1}^{N} p(\theta^k | x_{0:n}, z_{0:n}, u_{0:n}, \eta_{0:n}),$$

where θ^k, $k = 1, \ldots, N$ are the features of the map [26, 31]. This factorization makes it possible to update one feature at a time and thus reduces memory requirements. Moreover,

"online" SLAM will be considered here, i.e. the map and the robot's position are constructed recursively as data become available, using

$$\begin{aligned} p(x_{0:n}, \Theta | z_{0:n}, u_{0:n}, \eta_{0:n}) \quad &\propto \quad p(x_{0:n-1}, \Theta | z_{0:n-1}, u_{0:n-1}, \eta_{0:n-1}) \\ &\quad \times p(x_n | x_{n-1}, u_n) p(z_n | \theta_n, x_n, \eta_n), \end{aligned} \tag{24}$$

where, θ_n is the feature observed at time n. Alternatively, one can wait and collect all the data, and then assimilate this large data set in one sweep; such a "smoothing" approach is related to graph based SLAM (see e.g. [16]) but will not be discussed further in this paper.

For online probabilistic SLAM, one assumes that M weighted samples $\{X^j_{0:n-1}, \Theta^j, W^j_{n-1}\}$ approximate

$$p(x_{0:n-1}, \Theta | z_{0:n-1}, u_{0:n-1}, \eta_{0:n-1}),$$

at time $n-1$ and then updates the samples to time n by applying importance sampling to

$$p(x_n | x_{n-1}, u_n) p(z_n | \theta_n, x_n, \eta_n).$$

Here it is assumed that only one feature is observed at a time (which is realistic [26]), however the extension to observing multiple features simultaneously is straightforward.

Various importance functions have been considered and tested in the literature. For example, the fastSLAM algorithm [27] was shown to be problematic in many cases due to the poor distribution of its samples [26, 28]. The reason is that the samples are generated by the motion model and, thus, are not informed by the data. As a result, the samples are often unlikely with respect to the data. The fastSLAM 2.0 algorithm [26,28] ameliorates this problem by making use of an importance function that depends on the data. The algorithm was shown to outperform fastSLAM and was proven to converge (as the number of samples goes to infinity) to the true SLAM posterior for linear models. The convergence for nonlinear models is not well understood (because of the use of extended Kalman Filters (EKF) to track and construct the map). Here a new SLAM algorithm with implicit sampling is described. The algorithm converges for nonlinear models at a computational cost that is comparable to the cost of fastSLAM 2.0.

Online SLAM with implicit sampling requires that implicit sampling is applied to the M functions F^j (one per sample), whose variables are the position at time t_n, x_n, and the location of the feature θ_n, observed at time t_n:

$$F^j(x_n, \theta) = -\log p(x_n | X^j_{n-1}, u_n) p(z_n | \theta^j_n, x_n, \eta^j_n);$$

the position of the jth sample at time t_{n-1} is a parameter. As in MCL, one needs M functions here due to the recursive problem formulation. The minimization problems of implicit sampling are

$$\phi^j = \min_{x_n, \theta_n} F^j(x_n, \theta_n),$$

and samples $\{X^j_n, \theta^j_n\}$ are generated by solving the stochastic equations (1). One can use the same technique for solving these equations as described in section 2. Moreover, if the

feature θ_n is already known, the SLAM algorithm reduces to the MCL algorithm with implicit sampling described above.

Note that the memory requirements of SLAM with implicit sampling are linear in the number of features, because the algorithm makes use of the conditional independence of the features given the robot path, so that, at each step, only one feature needs to be considered. Moreover, SLAM with implicit sampling converges to the true SLAM posterior (under the usual assumptions about the supports of the importance function and target pdf [14]) as the number of samples goes to infinity; this convergence is independent of linearity assumptions about the model and data equations. Convergence for fastSLAM and fastSLAM 2.0 however can only be proven for linear Gaussian models [26].

4.2.1 Numerical experiments

The applicability and efficiency of SLAM with implicit sampling is demonstrated with numerical experiments that mimic the University Car Park data set [32]. Here speed and steering data of [32] are used, however the laser data are replaced by synthetic data, because the data in [32] are too sparse (in time) for successful online SLAM (even EKF SLAM, often viewed as a benchmark, does not give satisfactory results).

In these synthetic data experiments, a virtual laser scanner returns noisy range and bearing measurements of features in a half-circle with 15m radius. If the laser detects more than one feature, one is selected at random and returned to the SLAM algorithm. Since each synthetic data set is a random event, 50 synthetic data sets are used to compute the average errors for various SLAM algorithms. These errors are defined as the norm of the difference of the ground truth and the conditional mean computed with a SLAM algorithm. The performances of EKF SLAM, fastSLAM, fastSLAM 2.0 and SLAM with implicit sampling are compared using these synthetic data sets. All SLAM methods make use of the same maximum likelihood algorithm for the data association. A Newton method was implemented for the optimization in implicit sampling, and the quadratic approximation of F was used to solve the algebraic equations (1). A typical outcome of a numerical experiment is shown in the left panel of figure 7, where the true path and true positions of the landmarks as well as their reconstructions via SLAM with implicit sampling (100 samples) are plotted.

The results of 50 numerical experiments with fastSLAM, fastSLAM 2.0 and SLAM with implicit sampling are shown in table 7. It was observed in this example that EKF SLAM gives an average error of 3.89% at an average computation time of 0.43 ms and, thus, is computationally efficient and accurate. However, computational efficiency disappears in scenarios with larger maps due to the $O(N^2)$ memory requirements (N is the number of features in the map). The error of fastSLAM 2.0 (whose memory requirements scale linearly with N) decreases with the number of samples, and it seems as if the fastSLAM 2.0 solution converges (as the number of samples and, thus, the computation time increases) to the EKF solution. This is intuitive because fastSLAM 2.0 uses EKFs to track the map. The fastSLAM algorithm, on the other hand, converges to an error that is larger than in EKF SLAM or fastSLAM 2.0. SLAM with implicit sampling converges to an error that is smaller than in

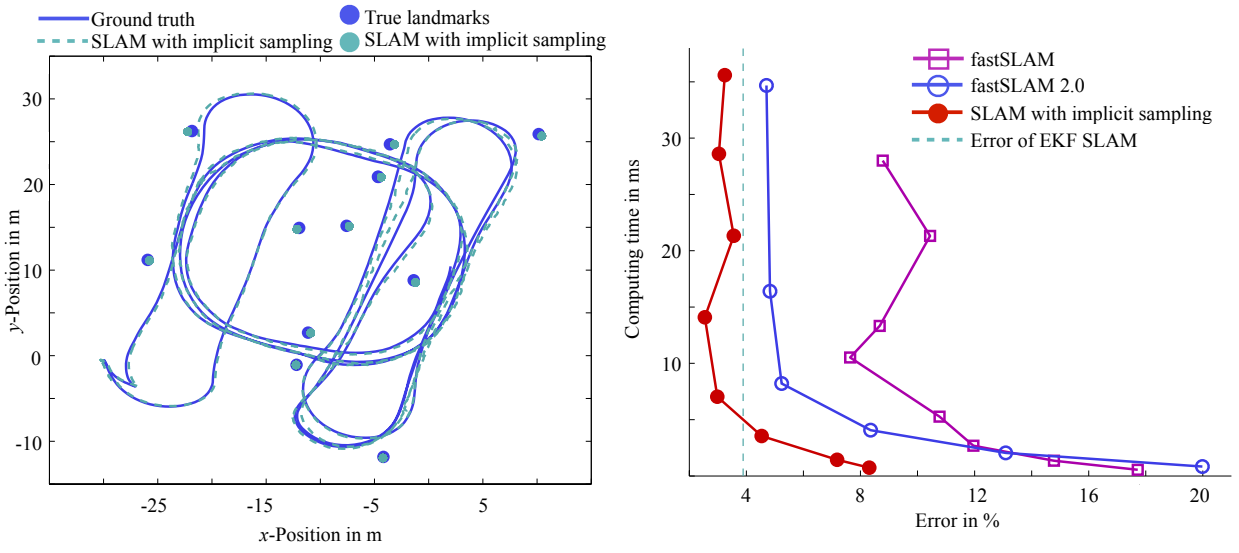

Figure 7: Simulation results for SLAM. Left: ground truth and reconstruction via SLAM with implicit sampling (100 samples). Right: computing time as a function of the average error of fastSLAM, fastSLAM 2.0 and SLAM with implicit sampling.

EKF SLAM. The reason is as follows: implicit sampling requires no linearity assumptions, and, therefore, the empirical estimate it produces converges (as the number of samples goes to infinity) to the true SLAM posterior.

Moreover, the convergence rate of SLAM with implicit sampling is faster than for any other SLAM method, due to the data informed importance function. Thus, the approximation of the conditional mean (the minimum mean square error estimator) one obtains with implicit sampling is accurate even if the number of samples is relatively small. As a result, SLAM with implicit sampling is more efficient than the other SLAM algorithms considered here. This is illustrated in the right panel of figure 7, where the computing time is shown as a function of the average error. As in MCL (see above), the area between the various curves indicates the improvement in efficiency. Here implicit sampling is the most efficient method, giving a high accuracy (small error) at a small computational cost. Moreover, SLAM with implicit sampling can achieve an accuracy which is higher than the accuracy of all other methods (including EKF SLAM), because it is a convergent algorithm.

5 Conclusions

Implicit sampling is a Monte Carlo scheme that localizes the high-probability region of the sample space via numerical optimization, and then produces samples in this region by solving algebraic equations with a stochastic right hand side. The computational cost per sample in implicit sampling is larger than in many other Monte Carlo sampling schemes that

Table 6: Simulation results for SLAM

	fastSLAM		fastSLAM 2.0		Implicit sampling	
Samples	CPU time in ms	Error in %	CPU time in ms	Error in %	CPU time in ms	Error in %
10	-	-	-	-	7.31	8.31
20	5.55	17.7	8.36	20.0	14.4	7.18
50	13.6	14.8	20.5	13.1	35.4	4.54
100	26.9	12.0	40.7	8.36	70.3	2.98
200	52.7	10.8	82.1	5.24	140.9	2.55
400	105.2	7.64	164.0	4.83	286.0	3.25
500	133.2	8.67	346.7	4.71	355.9	3.25
800	213.2	10.4	-	-	-	-
1000	280.1	8.78	-	-	-	-

randomly explore the sample space. However, the minimization directs the computational power towards the relevant region of the sample space so that implicit sampling often requires fewer samples than other MC methods. There is a trade-off between the cost-per-sample and the number of samples and this trade-off was studied in this paper in the context of three applications: Monte Carlo localization (MCL) and probabilistic online SLAM in robotics, as well as path integral control.

In path integral control one solves the stochastic Hamilton-Jacobi-Bellman (HJB) equation with a Monte Carlo solver. This has the advantage that no grid on the state space is needed and the method is therefore (in principle) applicable to control problems of relatively large dimension. Implicit sampling was applied in this context to speed up the Monte Carlo calculations. The applicability of this approach was demonstrated on two test problems. Path integral control was also used to find local minima in non-convex optimization problems.

An implementation of implicit sampling for MCL was tested and it was found that implicit sampling performs better than standard MCL (in terms of computing time and accuracy), especially if the data are accurate and the motion model is noisy (which is the case typically encountered in practice [40]). Similarly, implicit sampling for SLAM outperformed standard algorithms (fastSLAM and fastSLAM 2.0). Under mild assumptions, but for nonlinear models, SLAM with implicit sampling converges to the true SLAM posterior as the number of samples goes to infinity, at a computational cost that is linear in the number of features of the map.

Acknowledgements

I thank Professor Alexandre J. Chorin of UC Berkeley for many interesting technical discussions and for bringing path integral control to my attention. I thank Dr. Robert Saye of Lawrence Berkeley National Laboratory for help with proofreading this manuscript. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under contract DE-AC02005CH11231, and by the National Science Foundation under grant DMS-1217065.

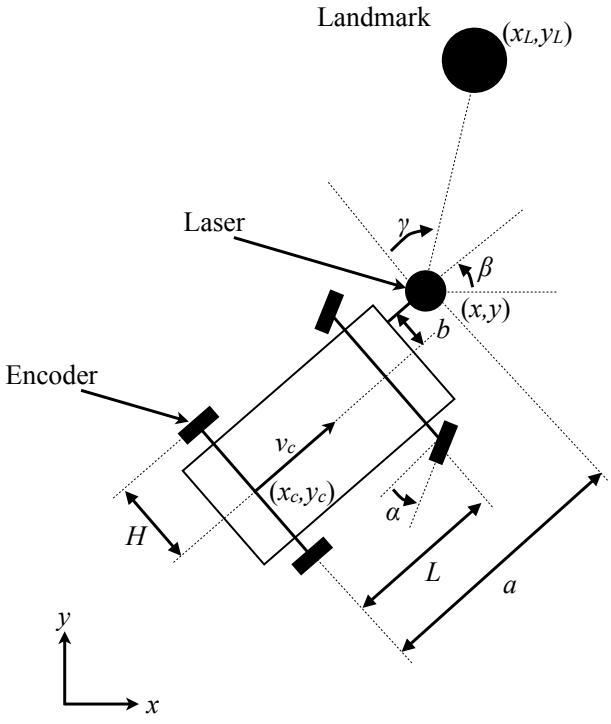

Figure 8: Vehicle kinematics (a version of Figure 1 in [17]).

Appendix

A kinematic model of the robot is described in [17] and shown for convenience in figure 8. The robot is controlled by the speed v_c and the steering angle α. It can be shown that the derivatives of the position and orientation of the axle of the robot are

$$\frac{dx_c}{dt} = v_c \cos(\beta), \quad \frac{dy_c}{dt} = v_c \sin(\beta), \quad \frac{d\beta}{dt} = \frac{v_c}{L} \tan(\alpha).$$

The velocity v_l however is measured at the rear left wheel and, thus, must be translated to the axle:

$$v_c = \frac{L v_l}{L - \tan\alpha H}.$$

The position of the laser can be obtained from the position of the axle by using

$$x = x_c + a\cos(\beta) - b\sin(\beta), \quad y = y_c + a\sin(\beta) + b\cos(\beta),$$

so that the forward model (without noise) is

$$\begin{aligned}\frac{dx}{dt} &= v_c\left(\cos(\beta) - \tan\alpha\left(a\sin(\beta) - b\cos(\beta)\right)\right),\\ \frac{dy}{dt} &= v_c\left(\sin(\beta) + \tan\alpha\left(a\cos(\beta) - b\sin(\beta)\right)\right),\\ \frac{d\beta}{dt} &= \frac{v_c}{L}\tan(\alpha),\end{aligned}$$

which, in a more compact notation, can be written as

$$\frac{dx}{dt} = R(x,u),$$

where $x = (x, y, \beta)^T$ and $u = (v_c, \alpha)^T$. The uncertainty in the model is due to errors in the inputs and in the model [17]:

$$\begin{aligned}dx &= R(x,\hat{u}(t))\,dt + Q\,dW_x,\\ \hat{u}(t)\,dt &= u(t)\,dt + P\,dW_u.\end{aligned}$$

where W_u and W_x are 2-, respectively 3-dimensional vectors whose components are independent Brownian motions, P and Q are real 2×2, respectively 3×3, matrices and $u(t)$ is the unperturbed control signal. The above equations are linearized in $\hat{u}$ to obtain

$$dx = R(x,u)\,dt + \frac{dR}{du}P\,dW_u + Q\,dW,$$

and the stochastic forward Euler method [24] with time step δ is used to discretize the equations:

$$x^{n+1} = x^n + R(x^n,u^n)\delta + \Delta W^n, \quad \Delta W^n = \mathcal{N}(0,\Sigma_1^n),$$

where

$$\Sigma_1^n = \delta\left(\frac{dR}{du}PP^T\frac{dR}{du}^T + QQ^T\right),$$

and $x^n = x(n\delta), u^n = u(n\delta)$. Table 7 lists the numerical values of all parameters of the probabilistic motion model. The steering data α and velocity data v_c are taken from the data set [32].

The data are measurements of range and bearing of the features in the map θ relative to the robot and are obtained by the laser. Only "high intensity points" from the data set [32] are used. The laser is modeled by the data equation

$$z^{j,n} = h(\hat{x}^n) + V^n, \quad V^n \sim \mathcal{N}(0,\Sigma_2),$$

where Σ_2 is a 2×2 diagonal matrix whose elements are $\sigma_1 = 0.05$ and $\sigma_2 = 0.05\pi/180$ and

$$h(x^n) = \begin{pmatrix} ||x^n - m^j|| \\ \operatorname{atan2}\left(\frac{m_2^n - \hat{x}_2^n}{m_1 - \hat{x}_1^n}\right) - x_3^n + \frac{\pi}{2}, \end{pmatrix},$$

Table 7: Model parameters.

Description	Parameter	Value
Wheel base	L	2.83m
Width of the car	H	0.76m
Horizontal offset of laser	b	0.5m
Rear axis to laser	a	3.78m
Time step	δ	0.025
Covariance of model noise	Q	$Q_{11} = 0.1$, $Q_{22} = 0.1$, $Q_{33} = 0.1\pi/180$ $Q_{ij} = 0$ if $i \neq j$
Covariance of noise in controls	P	$P_{11} = P_{22} = 0.5$, $P_{ij} = 0$ if $i \neq j$

where $\hat{x}_j^n$ is the jth element of the vector $\hat{x}^n$, the "true" robot position, and m_1^j, m_2^j are the x- and y-coordinates of the jth element of a landmark; the norm $||\cdot||$ is the Euclidean norm.

The gradient and Hessian of F_j in (22) are:

$$\nabla F_j = (\Sigma_1^n)^{-1}(x - f^n) + \sum_{i=1}^{p} \begin{pmatrix} \frac{x_1 - m_1^i}{\sigma_1 r}(r_i - z_1^i) + \frac{x_2 - m_2^i}{\sigma_2 r_i^2}(z_2^i - \hat{\theta}^i) \\ \frac{x_2 - m_2^i}{\sigma_1 r}(r_i - z_1^i) - \frac{x_1 - m_1^i}{\sigma_2 r_i^2}(z_2^i - \hat{\theta}^i) \\ \frac{z_2^i - \hat{\theta}^i}{\sigma_2} \end{pmatrix},$$

$$H_j = (\Sigma_1^n)^{-1} + \sum_{i=1}^{p} \hat{H}_i,$$

where the elements of the 3×3 matrices $\hat{H}^i$ are

$$\begin{aligned}
\hat{H}^i_{11} =& \frac{r_i - z^i_1}{\sigma_1 r_i} - \frac{(x_1 - m^i_1)^2 (r_i - z^i_1)}{\sigma_1 r_i^3} + \frac{x_1 - m^i_1}{\sigma_1 r_i^2} \\
&+ 2\frac{(x_2 - m^i_2)^3 (z^i_2 - \hat{\theta}^i)}{\sigma_2 (x_1 - m^i_1) r_i^4} - 2\frac{(x_2 - m^i_2)(z^i_2 - \hat{\theta}^i)}{\sigma_2 (x_2 - m^i_2) r_i^2)} + \frac{(x_2 - m^i_2)^2}{\sigma_2 r_i^4}, \\
\hat{H}^i_{12} = \hat{H}^i_{21} =& \frac{(x_1 - m^i_1)(x_2 - m^i_2)(r_i - z^i_1)}{\sigma_1 r_i^3} + \frac{(x_1 - m^i_1)(x_2 - m^i_2)}{\sigma_1 r_i^2} \\
&+ 2\frac{(x_2 - m^i_2)^2 (z^i_2 - \hat{\theta}^i)}{\sigma_2 r_i^4} + \frac{(z^i_2 - \hat{\theta}^i)}{\sigma_2 r_i^2} - \frac{(x_1 - m^i_1)(x_2 - m^i_2)}{\sigma_2 r_i^4}, \\
\hat{H}^i_{13} = \hat{H}^i_{31} =& \frac{x_2 - m^i_2}{\sigma_2 r_i^2}, \\
\hat{H}^i_{22} =& \frac{(x_2 - m^i_2)^2 (r_i - z^i_1)}{\sigma_1 r_i^3} + \frac{(r_i - z^i_1)}{\sigma_1 r_i} + \frac{(x_2 - m^i_2)^2}{\sigma_1 r_i^2} \\
&+ 2\frac{(x_1 - m^i_1)(x_2 - m^i_2)(z^i_2 - \hat{\theta}^i)}{\sigma_2 r^4} + \frac{(x_1 - m^i_1)^2}{\sigma_2 r^4}, \\
\hat{H}^i_{23} = \hat{H}^i_{32} =& \frac{x_1 - m^i_1}{\sigma_2 r_i^2}, \\
\hat{H}^i_{33} =& \frac{1}{\sigma_2},
\end{aligned}$$

and where

$$\begin{aligned}
r_i =& ||x^n - m^i||, \\
\hat{\theta}^i =& z^i_2 - \left(\text{atan2} \left(\frac{m^i - x_2}{m^i_1 - x_1} \right) - x_3 + \frac{\pi}{2} \right).
\end{aligned}$$

References

[1] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174 –188, 2002.

[2] E. Atkins, M. Morzfeld, and A.J. Chorin. Implicit particle methods and their connection with variational data assimilation. *Monthly Weather Review*, 141:1786–1803, 2013.

[3] J. Biswas, B. Coltin, and M. Veloso. Corrective gradient refinement for mobile robot localization. *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 73–78, 2011.

[4] J. Biswas and M. Veloso. Localization and navigation of the cobots over long-term deployments. *International Journal of Robotics Research*, 32(4):1679–1694, 2013.

[5] A.J. Chorin and O.H. Hald. *Stochastic Tools in Mathematics and Science*. Springer, third edition, 2013.

[6] A.J. Chorin, M. Morzfeld, and X. Tu. Implicit particle filters for data assimilation. *Communications in Applied Mathematics and Computational Science*, 5(2):221–240, 2010.

[7] A.J. Chorin and X. Tu. Implicit sampling for particle filters. *Proceedings of the National Academy of Sciences*, 106(41):17249–17254, 2009.

[8] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA '99)*, May 1999.

[9] G. Dissanayake, P. Newman, S. Clark, and H.F. Durrant-Whyte. A solution to the simultaneous localization and map building (SLAM) problem. In *IEEE Transactions of Robotics and Automation (ICRA '01)*, 2001.

[10] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[11] H.F. Durrant-Whyte. Uncertain geometry in robotics. In *IEEE Transactions of Robotics and Automation (ICRA '88)*, 1988.

[12] W.A. Fleming. Exit probabilities and optimal stochastic control. *Applied Mathematics and Optimization*, 4:329–346, 1977.

[13] R. Fletcher. *Practical Methods of Optimization*. Wiley, second edition, 1987.

[14] J. Geweke. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 24:1317 – 1399, 1989.

[15] M.P. Golombek, R.A. Cook, T. Economou, W.M. Folkner, A.F. Haldemann, P.H. Kallemeyn, J.M. Knudsen, R.M. Manning, H.J. Moore, T.J. Parker, R. Rieder, J.T. Schofield, P.H. Smith, and R.M. Vaughan. Overview of the Mars pathfinder mission and assessment of landing site predictions. *Science*, 5344(278):1743–1748, 1997.

[16] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[17] J.E. Guivant and E.M. Nebot. Optimization of simultaneous localization and map-building algorithm for real time implementation. *IEEE Transaction on robotics and automation*, 17(3):241 – 257, 2001.

[18] B.L Hammond, Jr. W.A. Lester, and P.J. Reynolds. *Monte Carlo Methods in ab initio Quantum Chemistry*. World Scientific Publishing, 1994.

[19] D. Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill, 1972.

[20] R.E. Kalman. A new approach to linear filtering and prediction theory. *Transactions of the ASME Journal of Basic Engineering*, 82(Series D):35–48, 1960.

[21] H.J. Kappen. Linear theory for control of nonlinear stochastic systems. *Physical review letters*, 95:200201, 2005.

[22] H.J. Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of statistical mechanics: theory and experiment*, 11:11011, 2005.

[23] H.J. Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. *AIP Conference Proceedings*, 2006.

[24] P.E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, third edition, 1999.

[25] D. Kortenkamp, R.P. Bonasso, and R. Murphy (editors). *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, 1998.

[26] M. Montemerlo and S. Thrun. *FastSLAM. A scalable method for the simultaneous localization and mapping problem in robotics*. Springer, 2007.

[27] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fast-SLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002.

[28] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[29] M. Morzfeld and A.J. Chorin. Implicit particle filtering for models with partial noise, and an application to geomagnetic data assimilation. *Nonlinear Processes in Geophysics*, 19:365–382, 2012.

[30] M. Morzfeld, X. Tu, E. Atkins, and A.J. Chorin. A random map implementation of implicit filters. *Journal of Computational Physics*, 231:2049–2066, 2012.

[31] K. Murphy. *Bayesian map learning in dynamic environments*. In: Advances in Neural Information Processing Systems, MIT Press, 1999.

[32] E.M. Nebot. University car park (inertial/gps) data set. http://www-personal.acfr.usyd.edu.au/nebot/car_park.htm, 2003. Last accessed in September 2014.

[33] E.M. Nebot. Ute documentation: Hardware manual. http://www-personal.acfr.usyd.edu.au/nebot/experimental_data/modeling_info/Ute_modeling_info.htm, 2003. Last accessed in September 2014.

[34] J. Nocedal and S.T. Wright. *Numerical Optimization*. Springer, second edition, 2006.

[35] J.-J.E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, 1991.

[36] R. Smith and P. Cheesman. On the representation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, 1987.

[37] R. Stengel. *Optimal Control and Estimation*. Dover, 1993.

[38] E.A. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11:3137–3181, 2010.

[39] C. Thorpe and H.F. Durrant-Whyte. *Field Robots*. ISRR, 2001.

[40] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[41] S. Thrun, D. Ferguson, D. Haehnel, M. Montemerlo, W. Burgard, and R. Triebel. A system for volumetric robotic mapping of abandoned mines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '03)*, 2003.

[42] S. Thrun, D. Fox, and W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31(1):29–53, 1998.

[43] E. Vanden-Eijnden and J. Weare. Rare event simulation and small noise diffusions. *Communications on Pure and Applied Mathematics*, 65:1770–1803, 2012.

[44] E. Vanden-Eijnden and J. Weare. Data assimilation in the low noise, accurate observation regime with application to the kuroshio current. *Monthly Weather Review*, 2013.

[45] B. Weir, R.N. Miller, and Y.H. Spitz. An implicit particle smoother for high-dimensional systems. *Nonlinear Processes in Geophysics*, 2013.

[46] I. Yang, M. Morzfeld, C.J. Tomlin, and A.J. Chorin. Path integral formulation of stochastic optimal control with generalized costs. *Proceedings of the 19th IFAC World Congress*, 2014.